##### **Лабораторные работы**

##### *по*

**DSP Builder**

# Software and Hardware Requirements to Complete All Exercises

**Software Requirements:** Quartus II; Matlab; DSP Builder; ModelSim-Altera

**Hardware Requirements:**  DSP Development Board

Лабораторная 1

# Лабораторная 1 – Использование Matlab Simulink для разработки алгоритма

## Цель работы:

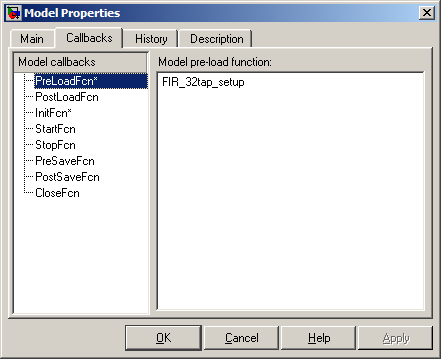
*Цель данной работы получить опыт использования Matlab / Simulink для разработки и верификации DSP алгоритмов. Вы разработаете FIR (КИХ) фильтр в Matlab, сгенерируете тестовые воздействия в Simulink модели при помощи Simulink блоков —и экспортируете эти воздействия в рабочую область (workspace) Matlab. После этого вы напишите сценарий Matlab для проверки производительности FIR фильтра.*

***Проект:***

*Тестовые воздействия будут созданы при помощи блоков Simulink. Входными данными для FIR фильтра будут либо сумма двух выборок синусоид (с частотами 0.01f/fs и 0.3 f/fs) либо белый шум. Суммированная синусоида будет подаваться в рабочую область Matlab, где FIR фильтр будет использован для режекции высокочастотной составляющей синусоиды. Промежуточные результаты будут отображаться при помощи блока осциллограф (scope) и спектральный осциллограф из библиотеки Simulink. Производительность FIR фильтра будет производиться в Matlab, при помощи сценария Matlab.*

**Шаг 1 – Создание входных воздействий в Simulink**

1. Откройте **MATLAB**. В окне **Current Directory**, установите путь к папке с проектными файлами **<exercise\_install\_directory>\Ex1\**.
2. Щёлкните на кнопку создания нового файла **New M-File**   для того чтобы открыть **M-File Editor**.
3. Напечатайте в открывшемся документе clock\_freq = 100e6;и sample\_time = 1/clock\_freq;(в две отдельные строчки). Сохраните файл под именем **FIR\_32tap\_setup.m** и закройте редактор.
4. Запустите **Simulink** щелчком по кнопке  . Откроется **Simulink Library Browser**.
5. Щёлкните на кнопку  (**File** -> **New** -> **Model**) для создания новой модели. Сохраните модель под именем **FIR\_32tap.mdl**.
6. Откройте окно свойств модели Model Properties (**File** -> **Model Properties**). Щелкните на вкладку Callbacks и напечатайте **FIR\_32tap\_setup** в разделе Model PreLoadFcn (как показано на картинке ниже). То же самое напечатайте в разделе InitFcn. Щёлкните OK и закройте окно Model Properties.



1. Из раздела **Source** библиотеки **Simulink**, перетащите в рабочее поле модели два блока **Sine Wave** (и назовите их Sine Wave 0.33fs и Sine Wave 0.01fs). Дважды щёлкните на блок для задания его параметров (все настройки, кроме указанных, остаются по умолчанию):

**Sine Wave 0.33fs**

**Sine type: Sample based**

**Amplitude: 2^9-1**

**Samples per period: 3**

**Sample time: sample\_time**

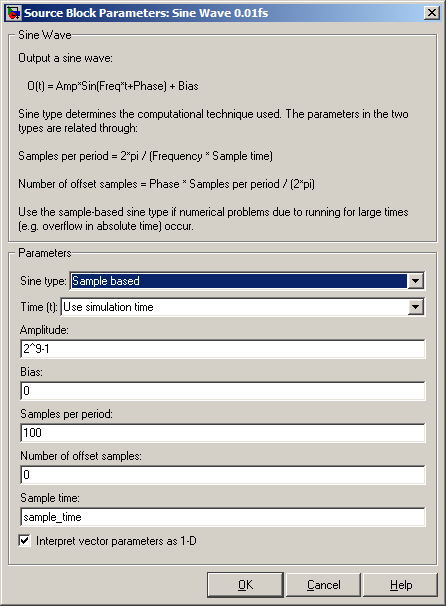
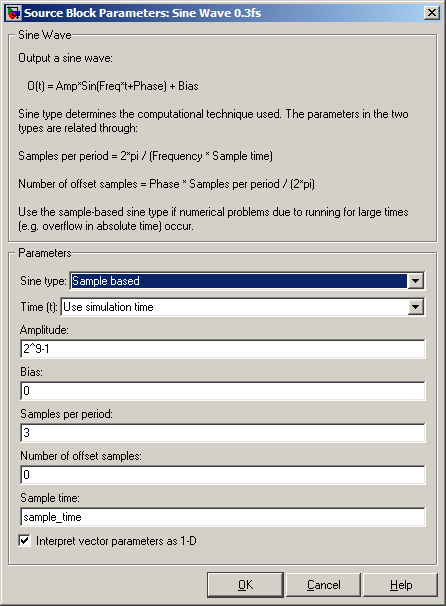
**Sine Wave 0.01fs**

**Sine type: Sample based**

**Amplitude: 2^9-1**

**Samples per period 100**

**Sample time: sample\_time**



Заметьте: f/fs = 1/(Samples per period)

1. Из раздела **Math Operations** библиотеки **Simulink**, добавьте блок **Add** в вашу Simulink модель.
2. Из раздела **Source** библиотеки **Simulink**, добавьте блок **Uniform Random Number** в вашу модель. Дважды щёлкните на него для задания следующих параметров (все, кроме указанных остаются по умолчанию):

**Uniform Random Number**

**Minimum: -2^10**

**Maximum: 2^10-1**

**Sample Time: sample\_time**

1. Из раздела **Signal Routing** библиотеки **Simulink**, добавьте блок **Manual Switch** в вашу Simulink модель.
2. Из раздела **Sinks** библиотеки **Simulink**, добавьте один блок **To Workspace** в модель **FIR\_32tap.mdl**. Дважды щёлкните на неё для задания следующих параметров (все, кроме указанных остаются по умолчанию):

**To Workspace**

**Variable name: wave\_in**

**Limit data points to the last: 256**

**Save format: Array**

1. Из раздела библиотеки **DSP System toolbox**, добавьте блок **Spectrum Scope** (назовите его Input Spectrum) в модель **FIR\_32tap.mdl**. Дважды щёлкните на неё для задания следующих параметров (все, кроме указанных остаются по умолчанию):

**Input Spectrum**

**Spectrum units Dbm**

**Buffer input: Checked**

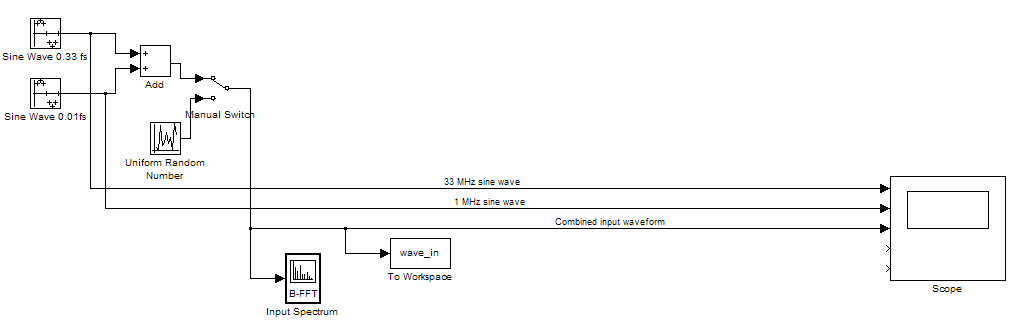
**Buffer size: 128**

**Buffer overlap: 64**

**Window type: Kaiser**

**Beta: 5**

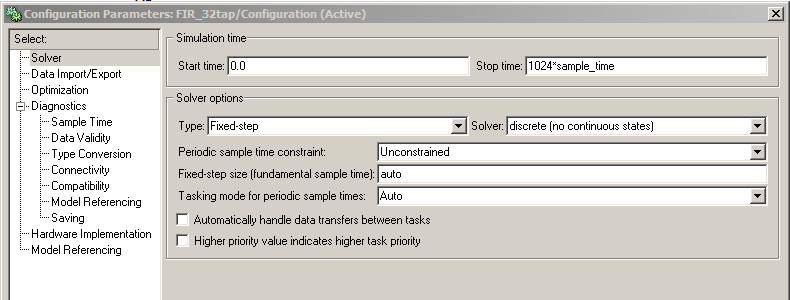
1. Из раздела **Sinks** библиотеки **Simulink**, добавьте блок **Scope** в модель **FIR\_32tap.mdl**. Дважды щёлкните на блок **Scope** и нажмите кнопку  (параметры **Scope**). На закладке **General**, поменяйте значение **Number of axes** на **5**. Выберите параметр **Sample Time** в разделе Sampling и измените его значение на **sample\_time**. Нажмите **OK**.
2. Соедините блоки в модели **FIR\_32tap.**mdl так как показано на рисунке:



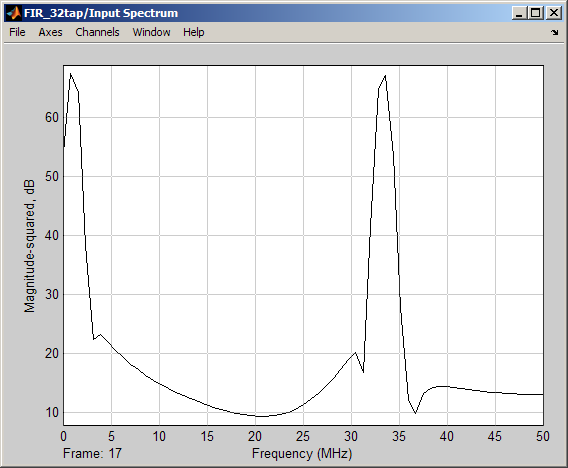
*Имя соединительной линии можно изменять, дважды щёлкнув на неё.*

**Шаг 2 – Моделирование в Simulink**

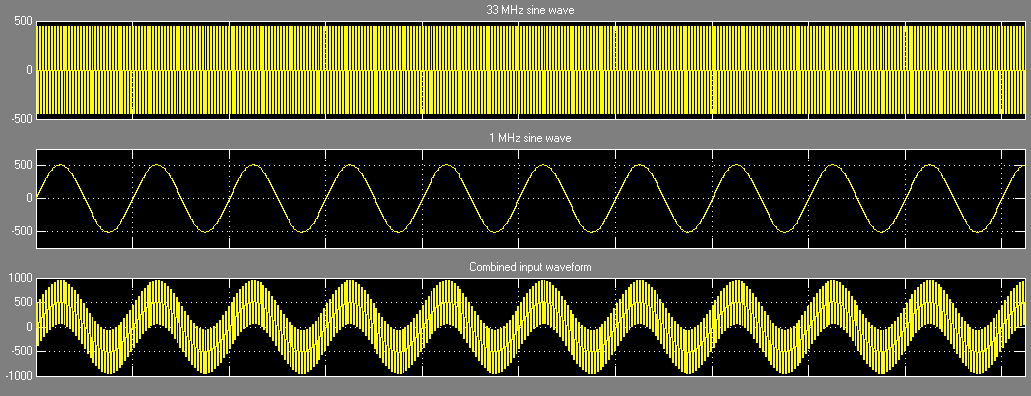
1. Из меню **Simulation**, Выберите **Configuration Parameters**.
2. Оставтье значение **Start** установленным в **0** и измените значение **Stop** на **1024\*sample\_time**.
3. В настройках **Solver options**, установите **Type** на **Fixed-Step** и **discrete (no continuous states**). Нажмите **OK**.



1. Дважды щёлкните на блок **Scope** чтобы открыть окно осциллографа. Щёлкните на кнопку  для запуска моделирования.
2. Окно осциллографа **Input Spectrum** откроется автоматически в процессе моделирования. По завершению моделирования вы должны увидеть следующий график:



1. Дважды щёлкните иконку Scope чтобы вызвать окно с графиками. Щёлкните на кнопку  для автоматической настройки масштаба осей. Графики должны выглядеть как на картинке ниже.



*В спектре входного сигнала вы должны видеть два пика: один на частоте ~1MHz (0.01fs) и один на частоте ~33MHz (0.33fs). Каждый пик кореллирует с синусоидой на соответствующей частоте.*

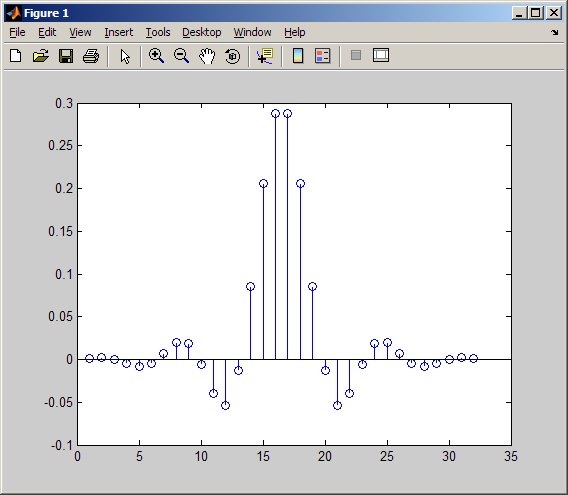
**Шаг 3 – Разработка и верификация FIR фильтра 32-го порядка в Matlab**

1. Используем алгоритм Parks-McClellan для разработки a 32 шагового фильтра равных пульсаций (equal ripple filter) в Matlab. В командной строке Matlab, введите **help firpm** для того чтобы ознакомится с функцией генерации коэффициентов.
2. Откройте файл **FIR\_32tap\_setup.m** и добавьте следующую строчку в конец файла:

**fir\_coeff = firpm(31,[0 0.1 0.2 0.5]\*2,[1 1 0 0]);**

Коротко опишите характеристики получившегося фильтра.

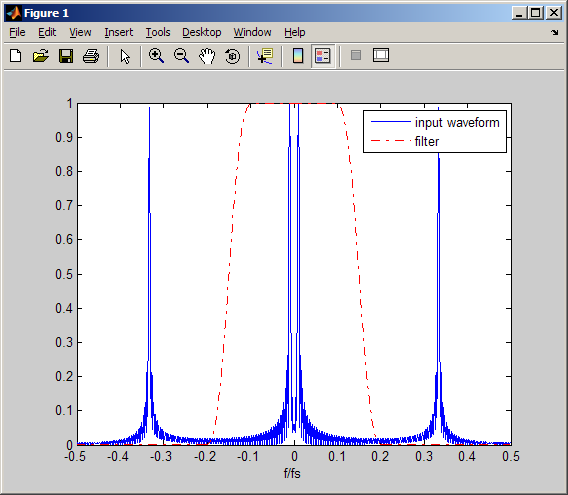
1. Не выходя из редкатора M-файлов, щёлкните кнопку  (Run) для того чтобы выполнить файл **FIR\_32tap\_setup.m** и запомнить коэффициенты фильтра в Matlab workspace
2. В командной строке Matlab, введите **stem(fir\_coeff)** для того, чтобы отобразить на графике коэффициенты фильтра. Вы должны увидеть следующую картинку:



*Обратите внимание - FIR фильтр это симметричный фильтр.*

1. Откройте прилагающийся **verify\_FIR.m** и ознакомьтесь с содержимым M-файла. Найдите определения функций (**help <function name>** в командной строке Matlab) и вкратце поясните что делает указанный сценарий.

1. Запустите сценарий **verify\_FIR.m** на исполнение, нажав на кнопку  (Run). Вы должны получить следующую картинку:



*FIR фильтр успешно обрезал высокочастотную составляющую.*

###### Обобщение

В данном упражнении на практике сделано следующие:

* + Использование M-файлов для параметризирования блоков Simulink
  + Использование блоков Simulink для генерации входных сигналов
  + Экспорт результатов моделирования Simulink в Matlab workspace
  + Проектирование FIR фильтра при помощи команды firpm в Matlab
  + Написание простейшего Matlab сценария для тестирования работоспособности FIR фильтра

Лабораторная 2

**Лабораторная 2- Использование DSP Builder для разработки аппаратуры**

## Цель работы:

*Целью данной работы является получение опыта разработки аппаратуры при помощи DSP Builder. Вы создадите FIR фильтр 32-го порядка, при помощи блоков из библиотеки Altera DSP Builder. Вы объедините разработанный FIR фильтр с различными блоками Simulink , разработанными в Лабораторной работе №1. Также вы научитесь генерировать VHDL код и компилировать его при помощи Quartus II.*

**Проект:**

*Используйте блоки из библиотеки DSP Builder для реализации FIR фильтра 32го порядка, изображенного на рисунке 2, в соответствии с инструкциями в этой работе. После этого фильтр и тестовые воздействия, разработанные в Лабораторной 1, моделируются в среде Simulink. В завершении VHDL код разработанного FIR фильтра генерируется при помощи DSP Builder и компилируется в пакете Quartus II.*

## 

**Рисунок 1**

## 

**Рисунок 2**

**Шаг 1 – Создание и параметризация FIR фильтра**

1. Скопируйте все свои файлы из директории **<exercise\_install\_directory>\Ex1\** в **<exercise\_install\_directory>\Ex2\**.
2. Откройте **FIR\_32tap\_setup.m** и добавьте в редакторе clock\_period = 1/(clock\_freq\*1e-9);после строки sample\_time = 1/clock\_freq. В конец файла добавьте строку fir\_coeff = round(fir\_coeff.\*(2^11-1)).

Ваш файл FIR\_32tap\_setup.m должен содержать следующие команды:

clock\_freq = 100e6;

sample\_time = 1/clock\_freq;

clock\_period = 1/(clock\_freq\*1e-9);

fir\_coeff = firpm(31,[0 0.1 0.2 0.5]\*2,[1 1 0 0]);

fir\_coeff = round(fir\_coeff.\*(2^11-1));

Сохраните **FIR\_32tap\_setup.m**,и закройте редактор.

1. Из раздела **AltLab** библиотеки **Altera** **DSP Builder**, добавьте блоки **SignalCompiler**, **Clock**, **Display Pipeline Depth** и **Resource Usage** в модель **FIR\_32tap.mdl**. Помните, что **SignalCompiler** это основной блок DSP Builder и он нужен для того чтобы перевести проект в HDL. Настройкте тактовую частоту **Clock** блока следующим образом:

**Clock**

**Real World Clock Period: clock\_period**

**Period Unit: ns**

**Simulink Sample Time: sample\_time**

1. Из раздела **IO & Bus** библиотеки **Altera DSP Builder blockset**, добавьте четыре блока **input**. Переименуйте их и параметризируйте как представлено в табличке ниже:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | **Parameters** | | | | |
| **Bus Type** | **[Number of Bits].[]** | **[].[Number of Bits]** | **Specify Clock** | **Clock** |
| ena | Single Bit |  |  | Checked | Clock |
| reset | Single Bit |  |  | Checked | Clock |
| data\_in | Signed Integer | 12 | 0 | Checked | Clock |
| in\_valid | Single Bit |  |  | Checked | Clock |

1. Из раздела **IO & Bus** библиотеки **Altera DSP Builder blockset**, добавьте три блока **output**. Переименуйте их и параметризируйте как представлено в табличке ниже:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Parameters** | | | |
| **Bus Type** | **[Number of Bits].[]** | **[].[Number of Bits]** | **External Type** |
| filtered\_out | Signed Fractional | 16 | 11 | Inferred |
| data\_out | Signed Integer | 12 | 0 | Inferred |
| out\_valid | Single Bit |  |  | Inferred |

1. Из раздела **Ports & Subsystems** библиотеки Simulink, добавьте блок **Subsystem** в ваш проект. Переименуйте блок в **32 tap FIR**. Дважды щёлкните на блок чтобы открыть его, и удалите линию связи между входом in1 и выходом out1. Дублируйте порт **in1** три раза и продублируйте порт **out1** два раза, чтобы создать три выходных и четыре входных порта. Переименуйте порты следующим образом:

Input ports: **in1**  **ena** **in2**  **reset** **in3**  **x\_in** **in4**  **in\_valid**

Output ports: **out1**  **y\_out** **out2**  **x\_out** **out3**  **out\_valid**

Закройте блок subsystem (вернемся к его редактированию позже).

*Блок* ***SubSystem*** *может использоваться для создания иерархии в проекте DSP Builder.*

1. Скопируйте блок **Input Spectrum** и назовите его **Output Spectrum**.
2. Из раздела **Sinks** библиотеки **Simulink**, добавьте один блок **Terminator** в файл **FIR\_32tap.mdl**.
3. Из раздела **Sources** библиотеки **Simulin**, добавьте блок **Step** в модель **FIR\_32tap.mdl**. Дважды щёлкните на него для задания следующих параметров (все, кроме указанных остаются по умолчанию):

**Step time: 1\*sample\_time**

**Sample time: sample\_time**

Дублируйте блок **Step** и создайте блок **Step1**.

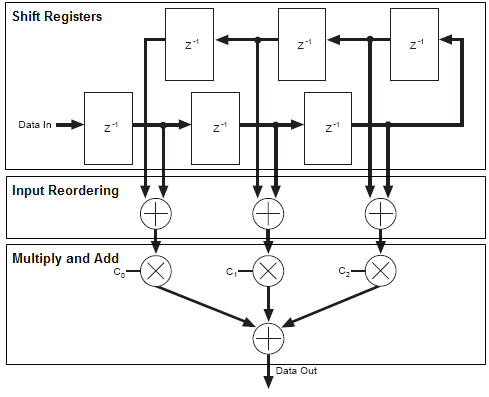
1. Из раздела **Sources** библиотеки **Simulink**, добавьте блок **Constant** в модель **FIR\_32tap.mdl**. Дважды щёлкните на него для задания следующих параметров (все, кроме указанных остаются по умолчанию):

**Constant value: 0**

**Sample time: sample\_time**

1. Соедините блоки так, как показа но на Рисунке 1.

*Таким образом, проект верхнего уровня завершен. Теперь приступим к проектированию FIR фильтра 32-го порядка. Воспользуемся тем, что FIR фильтр симметричен, и уменьшим его размер вполовину. Пример свертки в два раза фильтра 6-го порядка представлен на рисунке ниже.*

**

*Как представлено на рисунке выше, мы разобъём фильтр на 3 каскада: каскад сдвигающих регистров, каскад переупорядочивания входов и каскад умножения и сложения. Каскад сдвигающих регистров будет построен из сдвигающих блоков shift taps. Каскад умножений и сложений будет построен из комбинации блоков накапливающих умножителей (sum of product )и блоков параллельных сумматоров-вычитателей. Каскад переупорядочивания входов, будет построен в отдельной подсистеме,чтобы упростить структуру межсоединений FIR фильтра..*

*Приступим к построению сдвигающих регистров:*

1. Дважды щёлкните на блок **32 tap FIR** для того, чтобы открыть подсистему. Из раздела **Storage** библиотеки **Altera** **DSP Builder**, добавьте блок **Shift Taps** в подсистему **32 tap FIR**. Настройте блок **Shift Taps** следующим образом:

**Shift Tap**

*Main tab:*

**Number of Taps:** **8**

**Distance Between Taps: 1**

*Optional Ports and Settings tab:*

**Use Shift Out Port: checked**

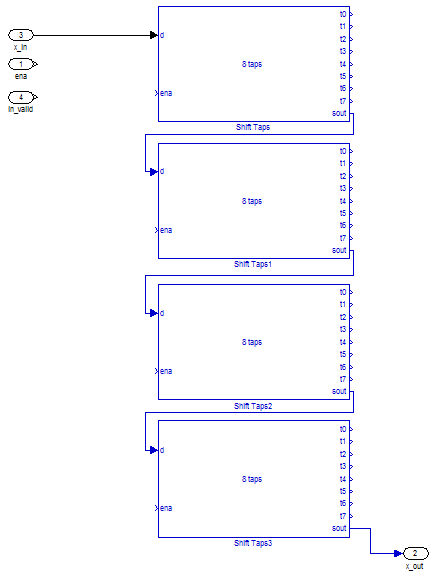
**Use Enable Port: checked**

**Use Dedicated Circuitry: checked**

**Memory Block Type: AUTO**

1. Дублируйте блок **Shift Taps** ещё 3 раза чтобы получить 4 блока **Shift Taps** (назовите их **Shift Taps**, **Shift Taps1**, **Shift Taps2** и **Shift Taps3**).
2. Соедините блоки **Shift Taps** следующим образом:
   * Соедините порт **sout** блока **Shift Taps** с портом **d** блока **Shift Taps1**.
   * Соедините порт **sout** блока **Shift Taps1** с портом **d** блока **Shift Taps2** block.
   * Соедините порт **sout** блока **Shift Taps2** с портом **d** блока **Shift Taps3** block.
   * Соедините порт **sout** блока **Shift Taps3** с портом **x\_out** (выходной порт 2).

*Ваш сдвигающий регистр должен выглядеть следующим образом:*

**

*Построение сдвигающих регистров закончено. Теперь разработаем блок сложения и умножения FIR фильтра.*

1. Из раздела **Arithmetic** библиотеки **Altera** **DSP Builder**, добавьте блок **Sum of Product** в подсистему **32 tap FIR**. Настройте блок **Sum of Product** следующим образом:

**Sum of Product**

*Main tab:*

**Input Data Number of Bits:****12**

**Number of Coefficients (1-8): 8**

**Coefficient Number of Bits: 12**

**Signed Integer Fixed-Coefficient Values (C\_i): fir\_coeff(1:8)**

**Number of Pipeline Stage: 1**

*Optional Port Sizing tab:*

**Full Resolution for Output Results: checked**

*Optional Ports and Settings tab:*

**FPGA Implementation:****Dedicated Multiplier Circuitry**

**Use Enable Port (ena): checked**

**Use Asynchronous clear port (aclr): checked**

1. Дублируйте блок **Sum of Product** и создайте аналогичный блок **Sum of Product1**. Все настройки остаются прежними, кроме коэффициентов (**C\_i**). Смените коэффициенты **C\_i** блока **Sum of Product1** на **fir\_coeff(9:16)**.
2. Из раздела **Arithmetic** библиотеки **Altera** **DSP Builder**, добавьте блок **Parallel Adder Subtractor** в подсистему **32 tap FIR**. Настройте блок **Parallel Adder Subtractor** следующим образом:

**Parallel Adder Subtractor**

*Main tab:*

**Number of Inputs:****2**

**Add (+) Sub (-): ++**

**Enable Pipeline (Variable Length): checked**

*Optional Ports tab:*

**Use Enable Port (ena): checked**

**Use Asynchronous clear port (aclr): checked**

1. Из раздела **I/O & Bus** библиотеки **Altera** **DSP Builder**, добавьте блок **Binary Point Casting** в подсистему **32 tap FIR**. Настройте блок **Binary Point Casting** следующим образом:

**Binary Point Casting**

**Bus Type:****Signed Fractional**

**Input [Number Of Bits].[]: 27**

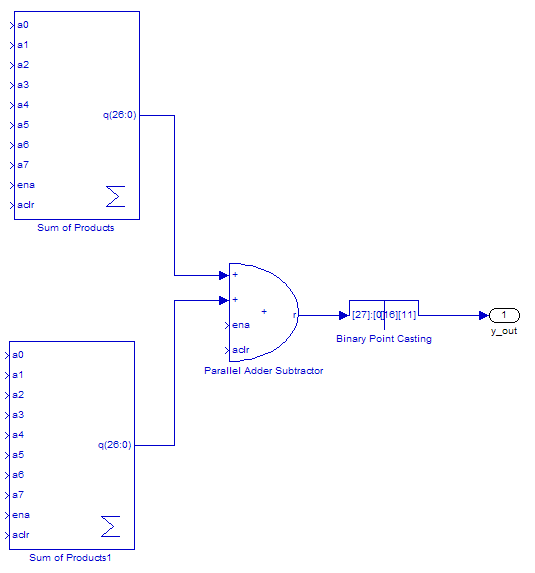
**Input [].[Number of Bits]: 0**

**Output Binary Position: 11**

*Коэффициенты FIR фильтра имеют динамический диапазон (-1,1). Поэтому нам необходим один знаковый бит и 11 значащих бит. Блок* ***Binary Point Casting*** *используется для масштабирования результатов, для того чтобы компенсировать рост разрядности после прохождения блока суммы произведений(sum of products).*

1. Соедините блоки **Sum of Products**, **Sum of Products1**,и **Parallel Adder Subtractor** следующим образом:
   * Соедините порты **q(26:0)** блока **Sum of Products** с одним из портов **+** блока **Parallel Adder Subtractor**.
   * Соедините порты **q(26:0)** блока **Sum of Products1** с другим портом **+** блока **Parallel Adder Subtractor**.
   * Соедините порт **r** блока **Sum of Products** с блоком **Binary Point Casting**.
   * Соедините выход блока **Binary Point Casting** с выходным портом **y\_out**

*Ваш умножитель-сумматор должен выглядеть следующим образом:*

**

*Проектирование сумматора-умножителя завершено. Приступим к проектированию каскада переупорядочивания входов.*

*Построение подсистемы переупорядочивания входов* ***Input Reordering*** *достаточно трудоёмко.*

1. Создайте новую подсистему (subsystem) (назовите её **Input Reordering**) внутри подсистемы **32 tap FIR**. Подсистема **Input Reordering** имеет 32 входных порта(названные **in1** - **in32**) и 16 выходных портов (называнные **out1** - **out16**).
2. Из раздела **Arithmetic** библиотеки **Altera** **DSP Builder**, добавьте блок **Pipeline Adder** в подсистему **Input Reordering**. Настройе блок **Pipeline Adder** следующим образом:

**Pipeline Adder**

*Main tab:*

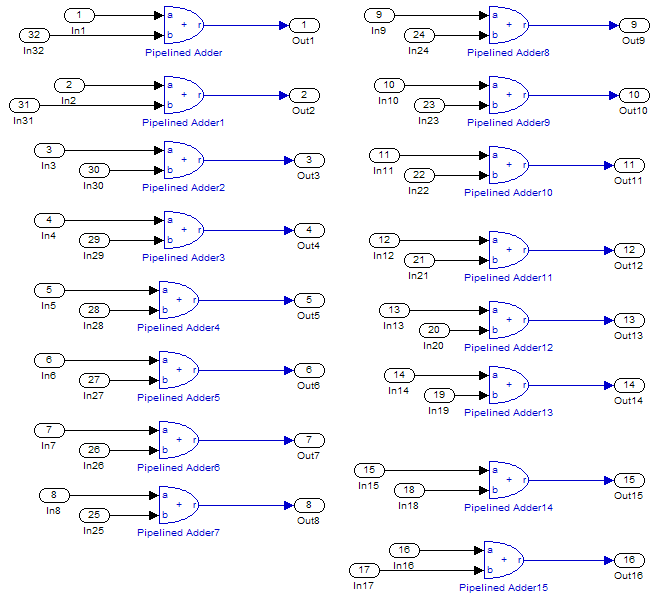
**Bus Type:****Signed Integer**

**[Number of Bits].[]: 12**

**[].[Number of Bits]: 0**

**Number of Pipeline Level: pipeline\_level**

1. Дублируйте блок **Pipeline Adder** ещё 15 раз, чтобы создать 16 копий блока **Pipeline Adder**. Соедините 32 входных порта **input ports** (**in1** – **in32**) и 16 выходных портов **output ports** (**out1** – **out 32**) с 16ю блоками **Pipeline Adder** (**Pipeline Adder** – **Pipeline Adder15**) следующим образом:

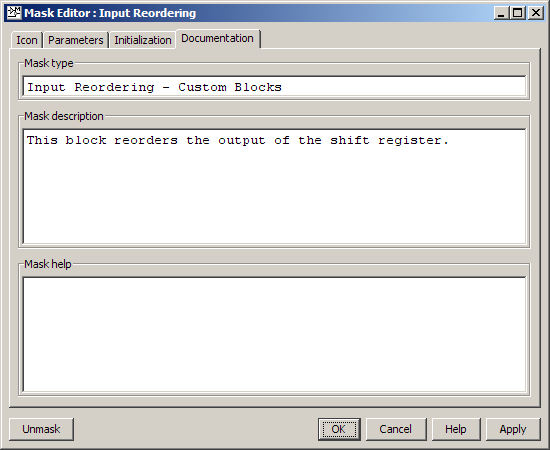


*Построение подсистемы* ***Input Reordering*** *завершено.Закройте подсистему и перейдите к подсистеме* ***32 tap FIR****. Следующие шаги сделают подсистему* ***Input Reordering*** *параметризируемой.*

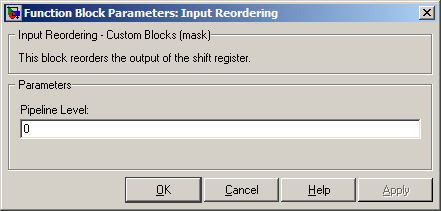
1. Щёлкните правой кнопкой на подсистему **Input Reordering** и выберите пункт меню **Mask Subsystem…** для того чтобы запустить редактор **Mask Editor**.
2. Щёлкните на закладку параметры для того чтобы открыть страницу с параметрами.
3. Щёлкните на кнопку  для того чтобы добавить параметр pipeline level. Заполните поля так, как показано на картинке ниже:



1. Щёлкните вкладку **Documentation** и заполните поля так, как показано на картинке ниже:



1. Щёлкните OK для закрытия редактора **Mask Editor**.
2. Дважды щёлкните на подсистему **Input Reordering** для того чтобы вызвать редактор **Function Block Parameters**. Задайте значение pipeline level так, как показано ниже:



*После того как подсистема замаскирована параметрами вы можете посмотреть содержимое подсистемы щёлкнув на неё правой кнопкой мышки и выбрав пункт “Look Under Mask”. Для того чтобы присвоить подсистеме дополнительные параметры, щёлкните на неё правой кнопкой мышки и выберите пункт “Edit Mask”.*

*.*

1. Из раздела **Storage** библиотеки **Altera** **DSP Builder**, добавьте блок **Delay** в подсистему **32 Tap FIR**. Настройте блок **Delay** следующим образомы:

**Delay**

*Main tab:*

**Number of Pipeline Stages:****35**

*Optional Ports tab:*

**Use Enable Port (ena): checked**

**Use Asynchronous clear port (aclr): checked**

1. Из раздела **Gate & Control** библиотеки **Altera** **DSP Builder**, добавьте блок **Logical Bit Operator** в подсистему **32 Tap FIR**. Настройте блок **Logical Bit Operator** следующим образом:

**Logical Bit Operator**

**Logical Operation:****AND**

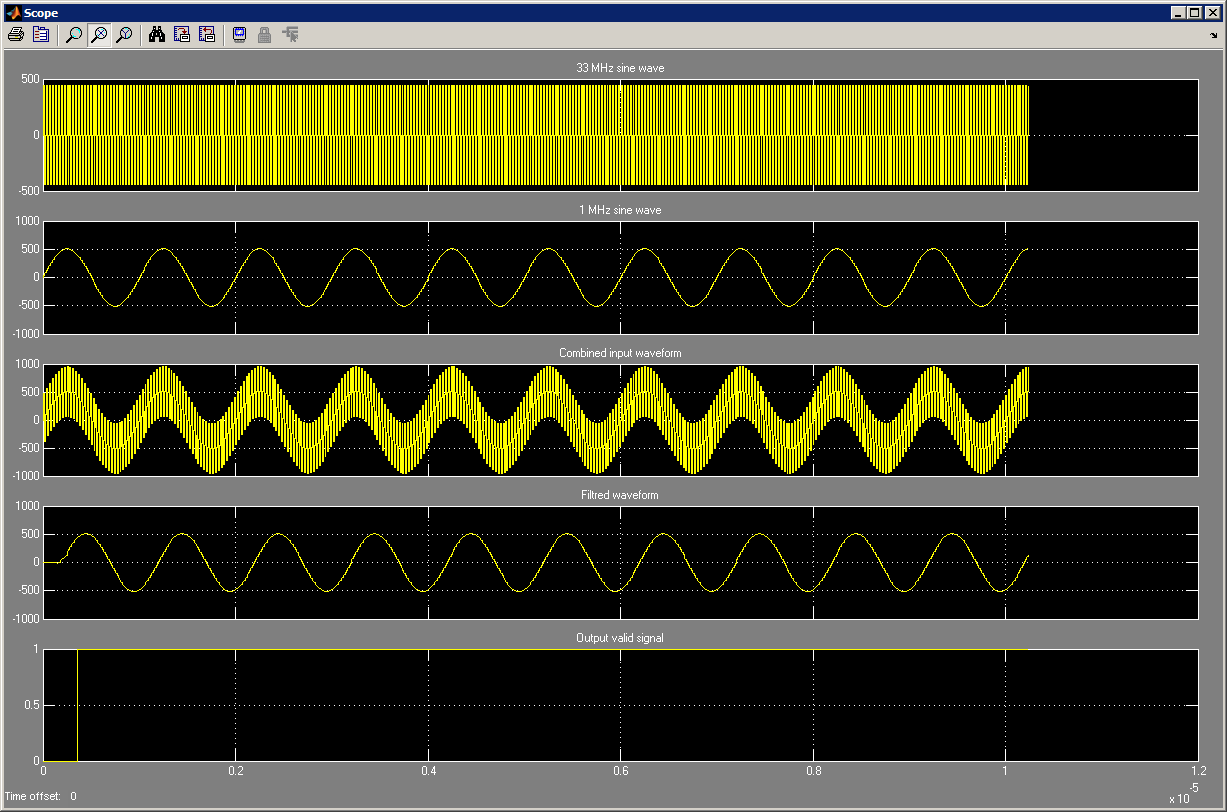
**Number of Inputs: 2**

1. Соедините все блок внутри подсистемы **32 Tap FIR** следующим образом:
   * Порты **t0** – **t7** блока **Shift Tap** с портами **in1** – **in8** подсистемы **Input Reordering**.
   * Порты **t0** – **t7** блока **Shift Tap1** с портами **in9** – **in16** подсистемы **Input Reordering**.
   * Порты **t0** – **t7** блока **Shift Tap2** с портами **in17** – **in24** подсистемы **Input Reordering**.
   * Порты **t0** – **t7** блока **Shift Tap3** с портами **in25** – **in32** подсистемы **Input Reordering**.
   * Соедините порт **reset** посистемы с входным портом(**port 2**) и с входами **aclr** блока **Sum of Products** и входом **sclr** блока **Delay**.
   * Соедините входы **ena** и **in\_valid** подсистемы (**port 1 и 4**) с входами элемента **AND**.
   * Соедините выход элемента **AND** со всеми ena портами (4 от блоков **Shift Taps**, два от блоков **Sum of Products**, один к блоку **Parallel Adder and Subtractor** и один к блоку **Delay**).
   * Соедините вход подсистемы **in\_valid** (**port 4**) с входным портом блока **Delay**.

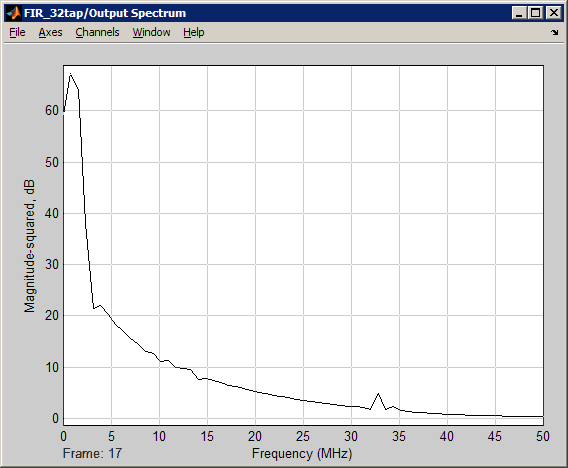
*Подсистема 32 Tap FIR должна выглядеть так как представлено на рисунке 2. Закройте подсистему и перейдите к файлу* ***FIR\_32tap.mdl****.*

**Шаг 2 – Моделирование в Simulink**

1. Щёлкните на кнопку  для запуска моделирования.
2. После того как моделирование завершено откройте окно осциллографа. Вы должны увидеть следующую картинку.

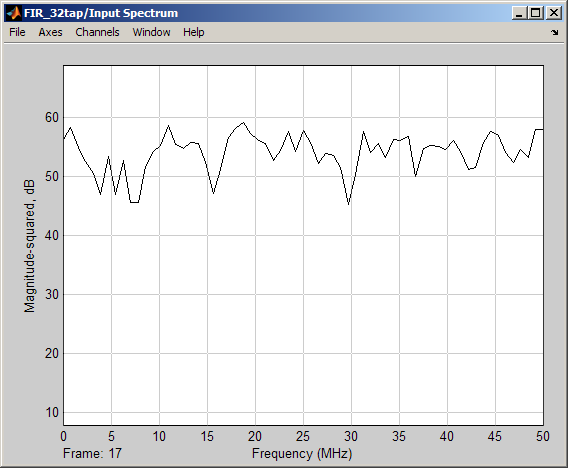
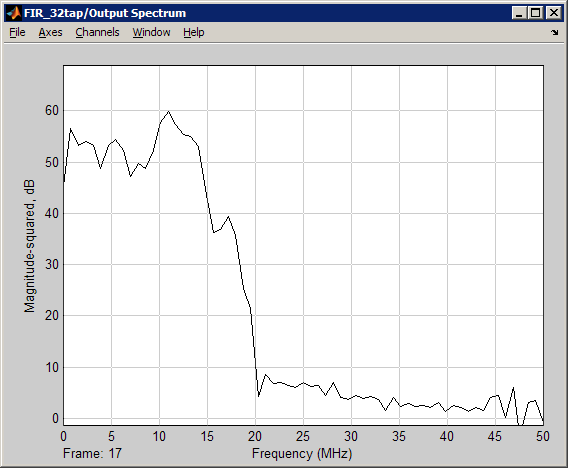


1. Также вы должны увидеть следующий спектр:



*В данном моделировании вы должны увидеть две оригинальные сэмплированные синусоиды на верхних осциллограммах, их сумма представлена на третей осциллограмме. Отфильтрованная синусоида представлена на четвертой осциллограмме (обратите внимание, что высокочастотная составляющая отфильтрована). Это также подтверждает спектр выходного сигнала. Наконец, сигнал valid, представлен на нижней осциллограмме.*

1. Дважды щёлкните на блок **Manual Switch** для того чтобы переключить источник входных данных. На вход фильтра будут подаваться случайные данные с блока **Uniform Random Number**. Запустите моделирование заново и пронаблюдайте входной и выходной спектры.

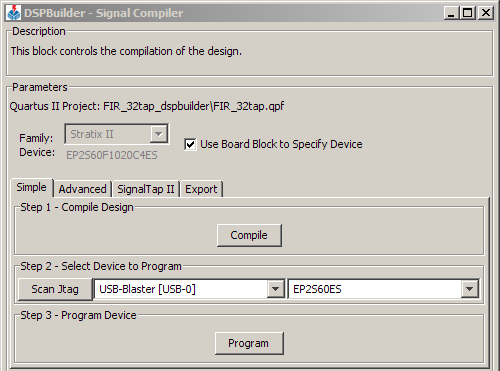
 

*Блок Uniform Random Number генерирует относительно плоский спектр сигнала примерно напоминающий белый Гауссов шум. Выходной спектр фильтра примерно похож на спектр идеального фильтра, созданного в Лабораторной №1.*

1. Дважды щёлкните на блок **Manual Switch** чтобы переключить входной сигнал обратно на сумму двух синуосоид.

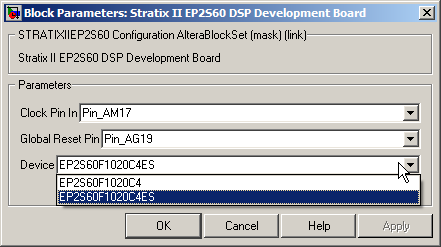
**Шаг 3 – Использование SignalCompiler для генерации VHDL, синтеза, и RTL моделирования**

1. Щёлкните на кнопку  чтобы раскрыть раздел **Boards** библиотеки **Altera DSP Builder**. Щёлкните на STRATIXIIEP2S60. Перенесите один блок **Stratix II EP2S60 DSP Development Board** в модель **FIR\_32tap.mdl**.
2. Дважды щёлкните на **SignalCompiler**. Удостоверьтесь в том, что установлена галочка на опции **Use Board Block to Specify Device**. Данная опция указывает, что информация о целевой микросхеме берётся из блока **Board Block (Stratix II EP2S60 DSP Development Board** в нашем случае**)**.



*Обратите внимание, некоторое платы прототипирования содержат микросхемы являющиеся инженерными образцами, а некоторые содержат серийные микросхемы. Следующие шаги позволят определить какой тип платы вы используете.*

1. Подсоедините USB Blaster к компьютеру. Подсоедините разъём JTAG USB Blaster к разъёму JTAG на плате прототипирования.
2. Включите питание платы.
3. В интерфейсе Signal Compiler, щёлкните кнопку Scan Jtag для того, чтобы определить какой тип. Если у вас инженерный образец, то сканирование Jtag вернёт значение **EP2S60ES** (как на скриншоте сверху). В противном случае сканирование Jtag вернёт **EP2S60**. Закройте Signal Compiler и выключите плату.
4. По умолчанию, блок **Stratix II EP2S60 DSP Development Board** указывает на серийную продукцию (EP2S60). Если у вас инженерный образец микросхемы, то вручную переключите значение Device на EP2S60ES.



1. Дважды щёлкните на **Signal Compiler** чтобы вызвать интерфейс Signal Compiler. Убедитесь в правильности выбора целевой микросхемы. Щёлкните кнопку **Compile**. Закройте **Signal Compiler** после завершения компиляции.
2. Дважды щёлкните на блок **Resource usage** для того чтобы посмотреть информацию о занятых ресурсах и результаты временного анализа.

*По результатам временного анализа видно, что наш FIR фильтр имеет проблемы с временными задержками (отрицательный период неполной загрузки (negative slack time)), особенно в цепи от сдвигающих регистров к блоку Sum of Products. Следующие шаги направлены на улучшение временных характеристик.*

1. Щёлкните на один из проблемных путей распространения сигнала и щёлкните на кнопку Highlight Path чтобы увидеть его на схеме.

*Инструмент автоматически откроет подсистему* ***32 tap FIR****. В зависимотри от того, какой из путей вы выбрали будут подсвечены либо блоки* ***Shift Taps*** *либо блоки* ***Sum of Products****.*

1. Дважды щёлкните на подсистему **Input Reordering** чтобы вызвать интерфейс **Functional Parameters**. Измените значение **pipeline level** на 1 и нажмите OK для завершения операции.

*Параметр pipeline level используется для контроля глубины конвейера сумматора входящего в состав подсистемы* ***Input Reordering****. При значении параметра pipeline level равном 0, подсистема Input Reordering строится полностью из на комбинаторной логике. Изменяя значение параметра pipeline level на 1, мы эффективно разрываем цепь между сдвигающими регистрами и блоком* ***Sum of Products*** *FIR фильтра.*

1. Перекомпилируйте проект и убедитесь в том, что отрицательный период неполной загрузки исчез из отчета временного анализатора.
2. Дважды щёлкните на блок **Display Pipeline Depth**, он должен изменить контекстную надпись на “*Display Depth Display On*”. Щёлкните на подсистему 32 Tap FIR и объясните почему мы выбрали задержку появления сигнала valid в 35 шагов конвейра (имейте ввиду, что мы предполагаем что первые 32 набора данных из FIR фильтра не действительны).

Обобщение лабораторной 2

Выполняя данную лабораторную работы вы научились:

* + Проектировать аппаратуру в среде Simulink при помощи блоков DSP Builder.
  + Генерировать VHDL код для проектов DSP Builder.
  + Компилировать проект в пакете Quartus II при помощи Signal Compiler.

Лабораторная 3

* **3A –Функциональное моделирование RTL при помощи блока TestBench**
* **3B – Ускорение моделирования при помощи Hardware In the Loop**

# Лабораторная 3A – Функциональное моделирование RTL при помощи блока TestBench

## Цель:

*Цель данной работы научится тестировать ваш DSP Builder проект при помощи моделирования RTL кода. Вы запустите функциональное моделирование фильтра,разработанного в предыдущей работе, в ModelSim. Результаты моделирования в ModelSim будут сравнены с результатами моделирования в Simulink.*

1. Скопируйте файлы из директории **<exercise\_install\_directory>\Ex2\** в директорию **<exercise\_install\_directory>\Ex3\parta**:
   * **FIR\_32tap.mdl**
   * **FIR\_32tap\_setup.m**
2. Откройте **MATLAB**, и установите рабочую директорию **Current Directory** на **<exercise\_install\_directory>\Ex3\parta**.
3. В редакторе моделей Simulink откройте файл **FIR\_32tap.mdl**.
4. Из раздела **AltLab** библиотеки**Altera** **DSP Builder**, добавьте блок **TestBench** в модель **FIR\_32tap.mdl**.
5. Дважды щёлкните на блок **TestBench** чтобы открыть интерфейс **Testbench** **generator**.
6. Установите галочку **Compare against HDL** для того, чтобы сравнить результаты моделирования в ModelSim и Simulink.

*По умолчанию, блок TestBench запустит моделирование в ModelSim без открытия графического интерфейса. Если вы хотите запустить графический интерфейс ModelSim, для того чтобы изучить временные диаграммы следует установить галочку* ***launch GUI*** *в настройках блока.*

*Вы должны увидеть то, что блок TestBench не сообщеает об ошибках или несовпадениях в моделировании.*

**Обобщение**

В данной части работы вы научились:

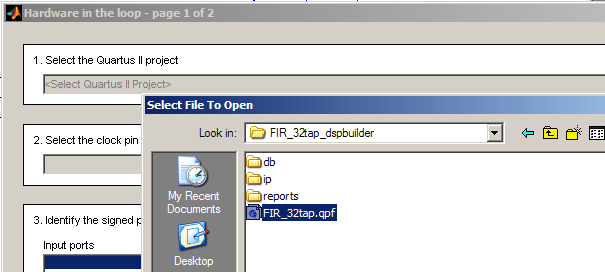
* + Запускать моделирование ModelSim из среды DSP Builder
  + Сравнивать результаты моделирования в Simulink и ModelSim используя блок TestBench

**Лабораторная 3B – Ускорение моделирования при помощи техники Hardware In the Loop**

## Цель:

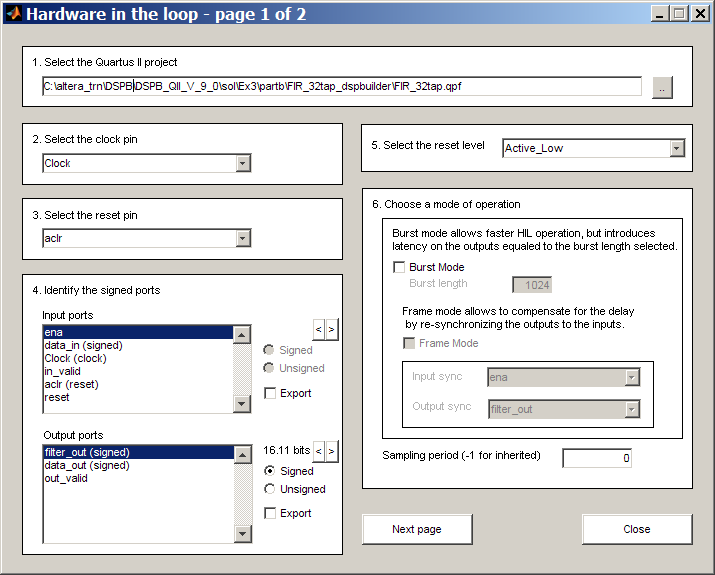
*Цель данной лабораторной научиться ускорять моделирование проекта при помощи технологии моделирования Hardware in the Loop*

1. Установите значение **Current Directory** на папку **<exercise\_install\_directory>\ex3\partb**.
2. Скопируйте следующие файлы из директории **<exercise\_install\_directory>\Ex3\parta** to **<exercise\_install\_directory>\Ex3\partb**:
   * **FIR\_32tap.mdl**
   * **FIR\_32tap\_setup.m**
3. Откройте **FIR\_32tap.mdl**.
4. Промоделируйте проект однократно и откомпилируйте его при помощи Signal Compiler. Закройте модель после завершения компиляции.
5. Откройте Simulink модель **FIR\_32tap\_HIL.mdl** в этой же директории
6. Соедините компьютер и плату прототипирования при помощи USB Blaster по интерфейсу JTAG.
7. Включите плату прототипирования.
8. Из раздела **AltLab** библиотеки **Altera** **DSP Builder** добавьте блок **HIL** в модель **FIR\_32tap\_HIL.mdl**.
9. Дважды щёлкните на блок **HIL.**
10. Выберите проект Quartus II с именем**FIR\_32tap.qpf** (в поддиректории **FIR\_32tap\_dspbuilder**)

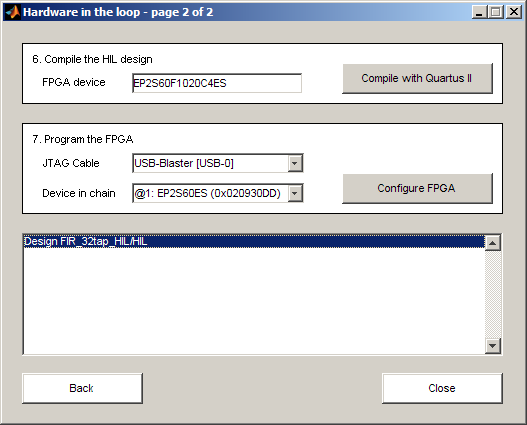


<Install directory>\

1. Установите для всех выходов outputs опцию **signed.** Нажимайте кнопку  до того, пока значение **filter\_out** не станет равно 16.11 бит.



1. Щёлкните **Next Page**
2. Пакет должен автоматически определить семейство FPGA и заполнить поля **FPGA device field** и **Device in chain** правильными значениями.



1. Щёлкните на кнопку **Compile with Quartus II** для запуска процесса компиляции в процессе которого будет пораждена специальная HIL версия вашего проекта.
2. Запрограммируйте FPGA нажав кнопку **Configure FPGA**.
3. Закйроте диалоговое окно HIL.
4. Соедините соответствующим образом выходы и выходы блоков **Input** и **Output** с блоком **HIL**. (Имейте ввиду, что некоторые входы могут соединятся напрямую).
5. Запустите моделирование в Simulink
6. Пронаблюдайте разницу в скоростях моделирования между HIL версией проекта и обычной версией проекта.

**Обобщение**

В этой части лабораторной вы:

* + Научились реализовывать HIL версию проекта для ускорения моделирования
  + Получили представление об уровне ускорения моделирования который обеспечивает блок HIL

# Лабораторная 3C – Тестирование аппаратуры при помощи встраиваемого логического анализаторы SignalTap II

## Цель:

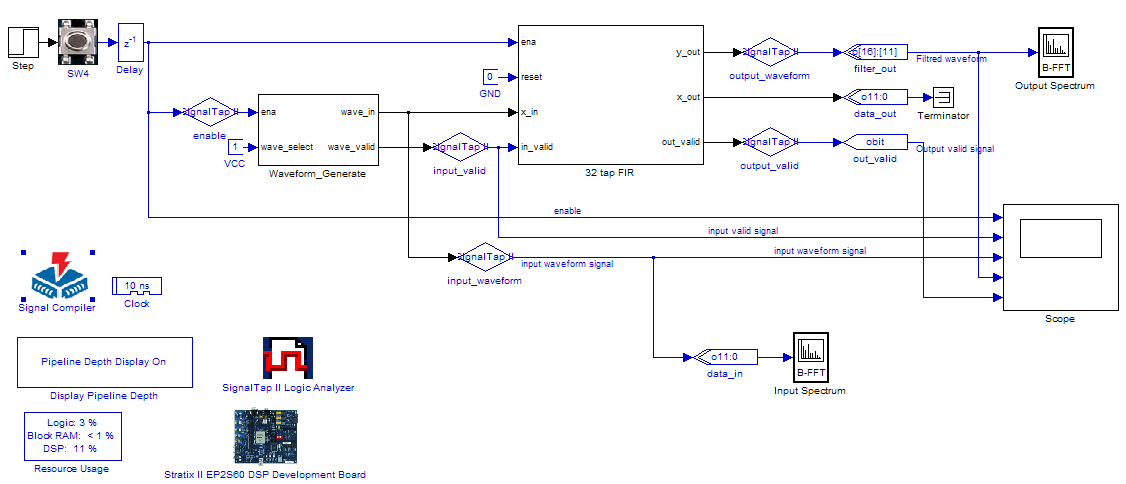
*Цель данной работы получить опыт исопльзования SignalTap II для оталдки разработанной аппаратуры.*

## Шаг 1 – Выбор точек (nodes) для анализа

1. В пакете MATLAB измените значение поля **Current Directory** на **<Lab\_Install\_Diretory>\ex3\partc** и откройте файл **FIR\_32tap.mdl.**

*Для того чтобы использовать SignalTap II Logic Analyzer для анализа разработанного FIR фильтра, входные тестовые сигналы должны генерироваться также в аппаратуре. Разработка блока генерации тестовых воздействий длительна, для экономии времени предоставляется готовый блок. Познакомьтесь с подсистемой* ***Waveform Generate*** *– она является генератором тестовых воздействий.*

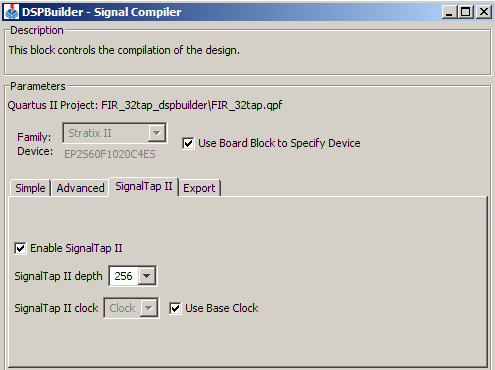
1. Откройте раздел AltLab в Simulink Library Browser. Перетащите блок **SignalTap II Node** в ваш проект. Спозиционируйте блок так, чтобы он был поверх соединительных линий после блока Delay и перед портом ena подсистемы **Waveform Generator**.
2. Щёлкните на текст под блоком и измените название сущности на **enable**
3. Повторите шаги 2-3, со следующими линиями:
   * Между поротом **wave\_valid** и портом **in\_valid**, с названием **input\_valid**
   * Между поротом **wave\_in** и выходным портом **data\_in**, с названием **input\_waveform**
   * Между поротом **y\_out** и выходным портом **filter\_out**, с названием **output\_waveform**
   * Между портом **out\_valid** и выходным портом **out\_valid** , с названием **output\_valid**
4. Из раздела **AltLib** библиотеки **Altera** **DSP Builder**, добавьте блок **Signal Tap II Logic Analyzer** в модель **FIR\_32tap.mdl**.
5. Сохраните файл. Модифицированная схема выглядит, так как представлено ниже

**

***Шаг 2 – Включение опции SignalTap II Option в Signal Compiler***

*Когда вы добавляете блоки типа node к сигналам, каждый блок по умолчанию подключается к встраиваемому логическому анализатору SignalTap II. Это функциональное изменение в проекте, поэтому необходимо перекомпилировать проект перед тем как использовать SignalTap II. Для перекомпиляции выполните следующие шаги:*

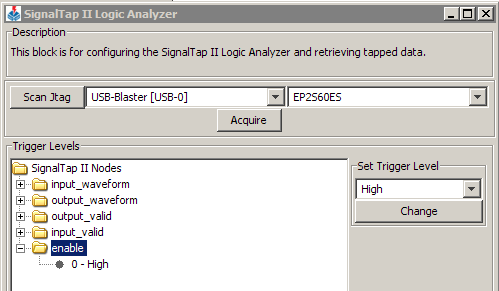
1. Дважды щёлкните на блок Signal Compiler block и сконфигурируйте вкладку **SignalTap II** следующим образом



1. Откомпилируйте проект на вкладке **Simple**
2. Запрограммируйте плату прототипирования

***Шаг 3 – Задание триггерных условий***

1. Дважды щёлкните на блок SignalTap II Logic Analyzer. В диалоговом окне будут отображены все nodes подключенные к SignalTap II для анализа



1. Укажите триггерное условие для нода **enable**:
   * Щёлкните на **enable** в списке Signal Tap II Nodes.
   * Выберите **High** в выпадающем списке **Set Trigger Level**.
   * Щёлкните кнопку **Change** чтобы изменения вступили в силу.

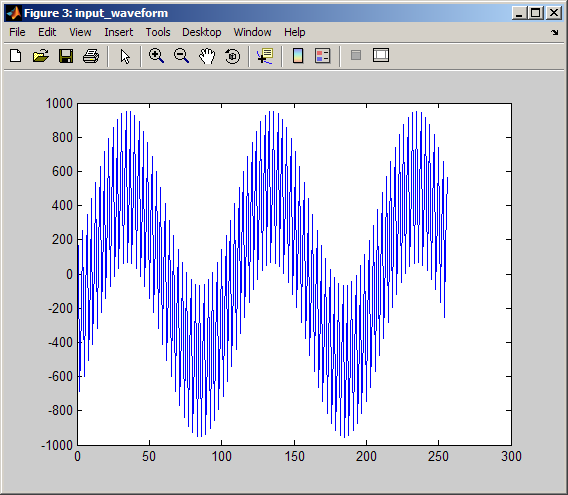
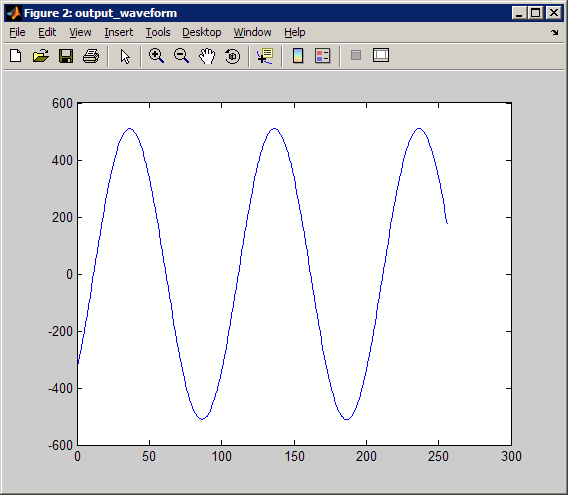
*Встраиваемый логический анализатор SignalTap II начинает захват данных для анализа тогда, когда обнаружит совместное срабатывание всех триггерных условий. В нашем случае SignalTap II сработает, когда обнаружит высокий уровень сигнала на линии enable.*

***Шаг 5 – Анализ при помощи SignalTap II***

1. Щёлкните **Scan Jtag** в диалоговом окне SignalTap II Analyzer и выберите соответствующий загрузочный кабель и целевую микросхему
2. Щёлкните **Acquire**
3. Нажмите переключатель SW4 на плате прототипирования (и удерживайте его несколько секунд) для того чтобы запустить триггерное событие для SignalTap II
4. Щёлкните **OK** в диалоговом окне SignalTap II Analysis когда вы закончите.

*Сохраненные данные отображаются на графика MATLAB и сохраняются в файла MATLAB с расширением .****mat*** *в рабочей директории.*

1. Проанализируйте графики входного и выходного сигналов в MATLAB

Обобщение работ 3A, 3B, 3C

При выполнении этой работы вы получили следующие навыки:

* + Использовать моделирование в ModelSim при помощи блока TestBench.
  + Сравнивать результаты моделирования в Simulink и ModelSim
  + Ускорять моделирование при помощи техники Hardware In the Loop
  + Проводить тестирование аппаратуры при помощи SignalTap II Embedded Logic Analyzer

Лабораторная 4

* **Интеграция модулей IP и системная отладка**

# Лабораторная 4 – Интеграция IP модулей и системная верификация

## Цель:

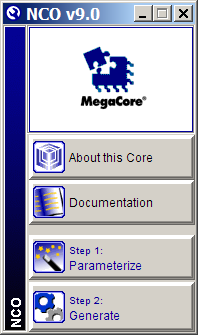
*Целью данной работы является получение навыков интегрирования мегафункций Altera IP в существующий проект DSP Builder. Генератор входных сигналов, реализованный на блоках Simulink, будет заменён на мегафункцию Altera Numeric Controlled Oscillator (NCO). Также изучаются вопросы тестирования всей системы при помощи сценариев Matlab M-file.*

***Шаг 1: Реализация и интеграция IP модуля NCO***

1. Откройте **MATLAB**. Установите значение **Current Directory** на путь **<exercise\_install\_directory>\Ex4**.
2. Откройте модель **FIR\_32tap.mdl**.

*Вместо того, чтобы использовать блоки Simulink или таблицы поиска (look up tables) (построенные из блоков Altera DSP) для генерации входных синусоид, построим часть проекта, отвечающего за генерацию(подсистема* ***Waveform Generator****) используя мегафункцию Altera Numeric Controlled Oscillator (NCO).*

1. Дважды щёлкните на подсистему **Waveform Generator**.
2. Из раздела **MegaCore Functions** библиотеки **Altera** **DSP Builder**, добавьте две сущности блоков **nco\_v9\_0** в подсистему **Waveform Generator**. Назовите один **high\_frequency\_NCO**, второй назовите **low\_frequency\_NCO**.
3. Дважды щёлкните на **high\_frequency\_NCO** чтобы открыть диалоговое окно задания свойств.



1. Щёлкните на **Step1: Parameterize** для того чтобы задать настройки **high\_frequency\_NCO**. Настройте **high\_frequency\_NCO** следующим образом:

**Delay**

*Parameters tab:*

**General Algorithm:****Multiplier-Based**

**Phase Accumulator Precision: 16**

**Angular Precision: 16**

**Magnitude Precision: 10**

**Implement Phase Dithering: checked**

**Dither Level: 4th from the Min side**

**Clock Rate: 100 MHz**

**Desired Output Frequency: 30 MHz**

*Implementation tab:*

**Frequency Modulation Input: unchecked**

**Phase Modulation Input: unchecked**

**Outputs: Single Output**

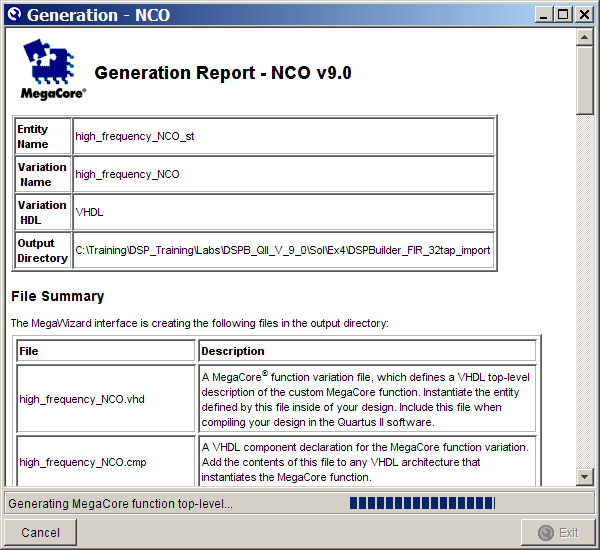
**Target: Stratix II**

**Number of Channels: 1**

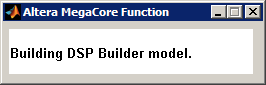
**Multiplier-Based Architecture: Use Dedicated Multiplier(s)**

**Clock Cycles Per Output: 1**

1. Щёлкните **Finish** для того чтобы закрыть окно с параметрами.
2. Щёлкниет на кнопку **Generate** чтобы сгенерировать код для **high\_frequency\_NCO**. Вы должны увидеть диалоговое окно генерации.



1. Обратите внимание на отчет о генерации (Generation Report) после того как MegaCore Function Generation сообщит об успешной генерации. Щёлкните на Exit чтобы закрыть окно.
2. После этого вы увидите всплывающее окно Altera MegaCore Function.



*Окно закроется автоматически, после того как DSP Builder закончит генерацию симуляционной модели для Simulink.*

1. Настройте **low\_frequency\_NCO** следующим образом:

**Delay**

*Parameters tab:*

**General Algorithm:****Multiplier-Based**

**Phase Accumulator Precision: 16**

**Angular Precision: 16**

**Magnitude Precision: 10**

**Implement Phase Dithering: checked**

**Dither Level: 4th from the Min side**

**Clock Rate: 100 MHz**

**Desired Output Frequency: 1 MHz**

*Implementation tab:*

**Frequency Modulation Input: unchecked**

**Phase Modulation Input: unchecked**

**Outputs: Single Output**

**Target: Stratix II**

**Number of Channels: 1**

**Multiplier-Based Architecture: Use Dedicated Multiplier(s)**

**Clock Cycles Per Output: 1**

1. Сгенерируйте код **low\_frequency\_NCO**.
2. Из раздела **Arithmetic** библиотеки **Altera** **DSP Builder**, добавьте блок **Parallel Adder Subtractor** в подсистему **Waveform Generator**. Настройте блок **Parallel Adder Subtractor** следующим образом:

**Parallel Adder Subtractor**

*Main tab:*

**Number of Inputs:****2**

**Add (+) Sub (-): ++**

**Enable Pipeline (Variable Length): unchecked**

1. Из раздела **IO & Bus** библиотеки **Altera** **DSP Builder**, добавьте два блока **Constant** в подсистему **Waveform Generator**. Назовите их **Constant** и **Constant1**. Настройте блоки **Constant** следующим образом:

**Constant**

**Constant Value:****phi\_inc\_l**

**Bus Type: Unsigned Integer**

**[Number Of Bits].[]: 16**

**Rounding Method: Truncate**

**Saturation Mode: Wrap**

**Specify Clock: checked**

**Clock: Clock**

**Constant1**

**Constant Value:****phi\_inc\_h**

**Bus Type: Unsigned Integer**

**[Number Of Bits].[]: 16**

**Rounding Method: Truncate**

**Saturation Mode: Wrap**

**Specify Clock: checked**

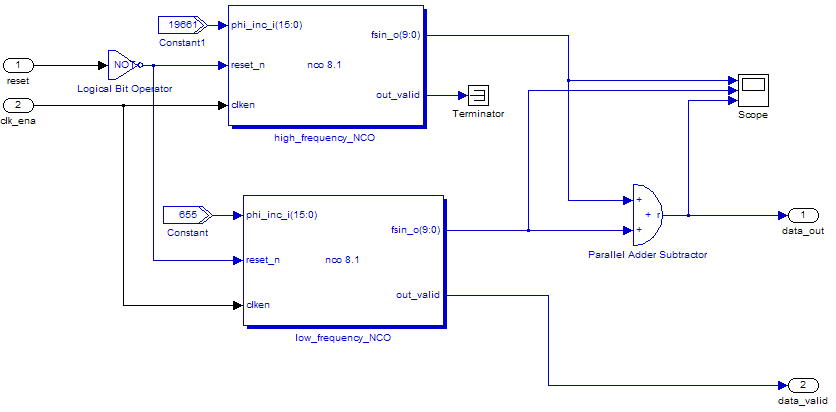
**Clock: Clock**

1. Из раздела **Gate & Control** библиотеки **Altera** **DSP Builder**, добавьте два блока **Logical Bit Operator** в подсистему **32 Tap FIR**. Настройте блок **Logical Bit Operator** следующим образом:

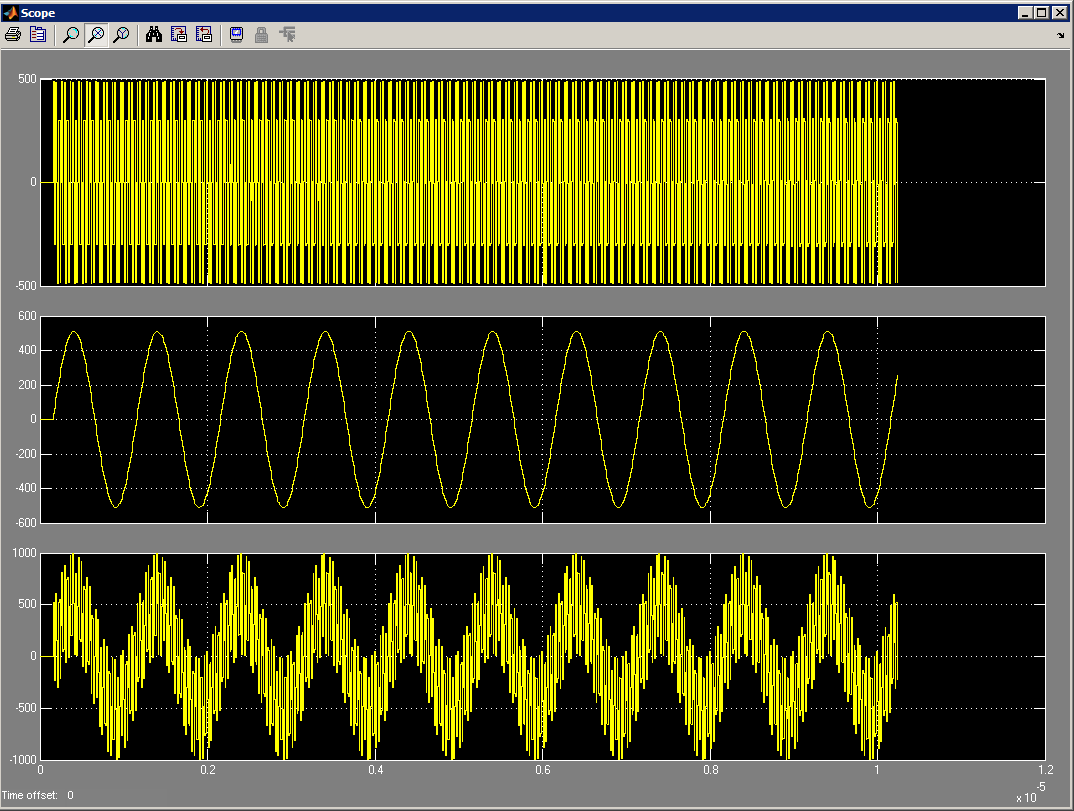
**Logical Bit Operator**

**Logical Operation:****NOT**

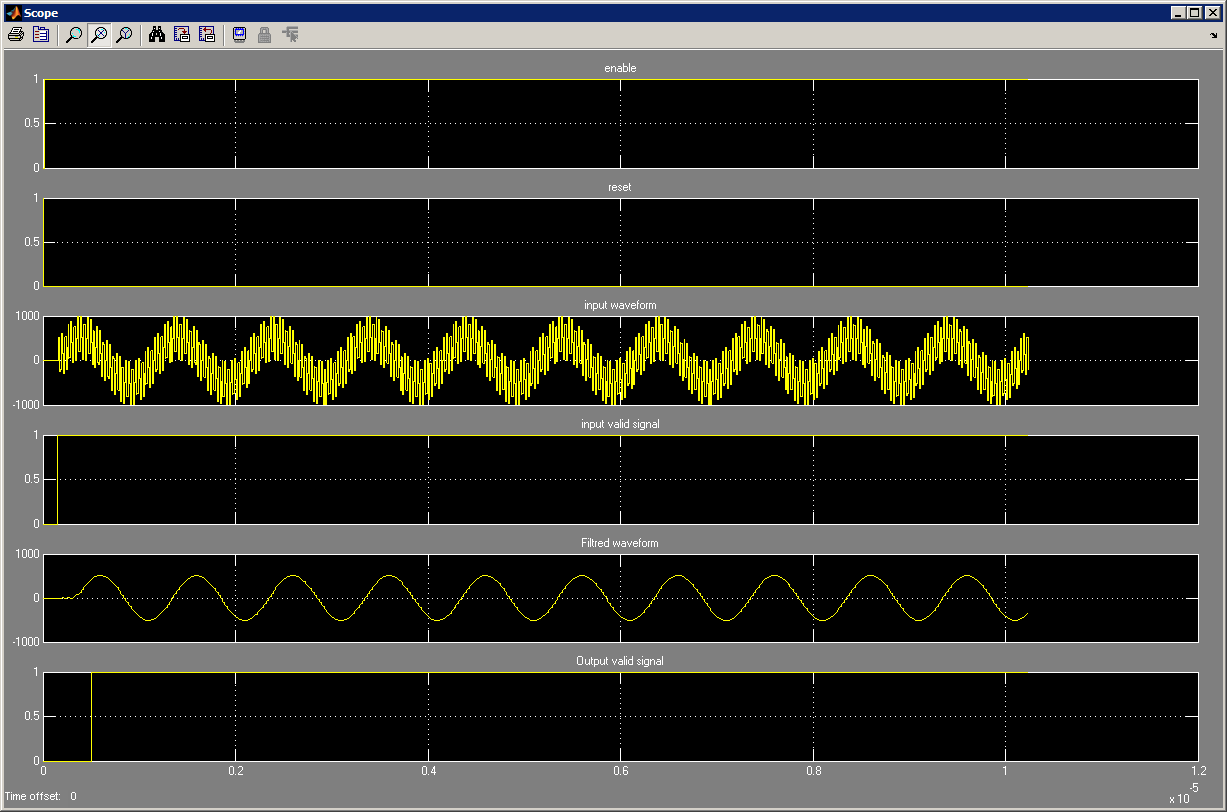
1. Из раздела **Sinks** библиотеки **Simulink**, добавьте блок **Scope** в подсистему **Waveform Generator**. Измените число портов в блоке **Scope** на три (3).
2. Из раздела **Sinks** библиотеки **Simulink**, добавьте блок **Terminator** в подсистему **Waveform Generator**.
3. Соедините блоки так, как показано на рисунке:



1. Промоделируйте проект. Вы должны увидеть следующие временные диаграммы:



Осциллограмма моделирования подсистемы **Waveform Generator**



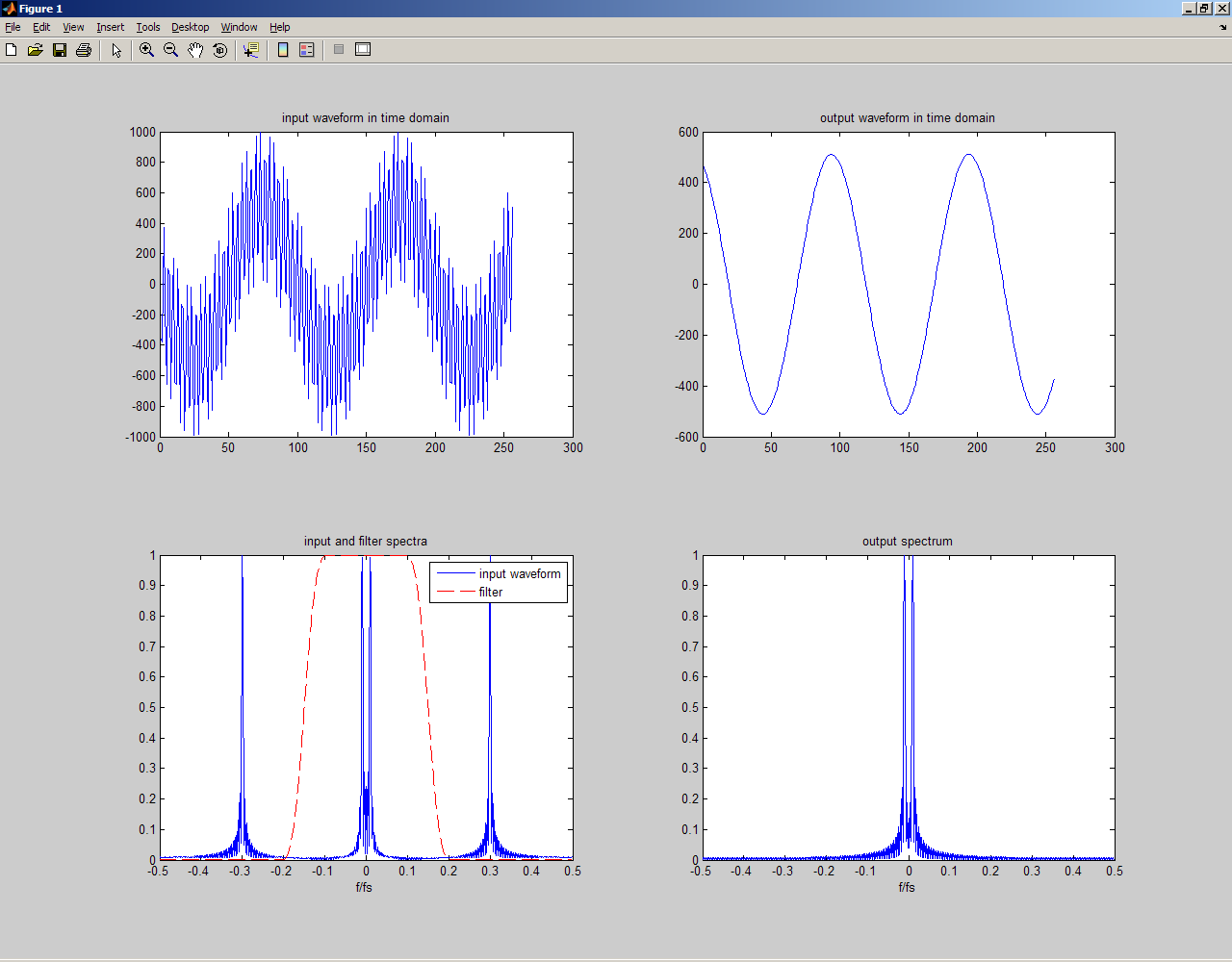
Осциллограмма моделирования системы **FIR\_32tap.mdl**

1. Закройте **FIR\_32tap.mdl**

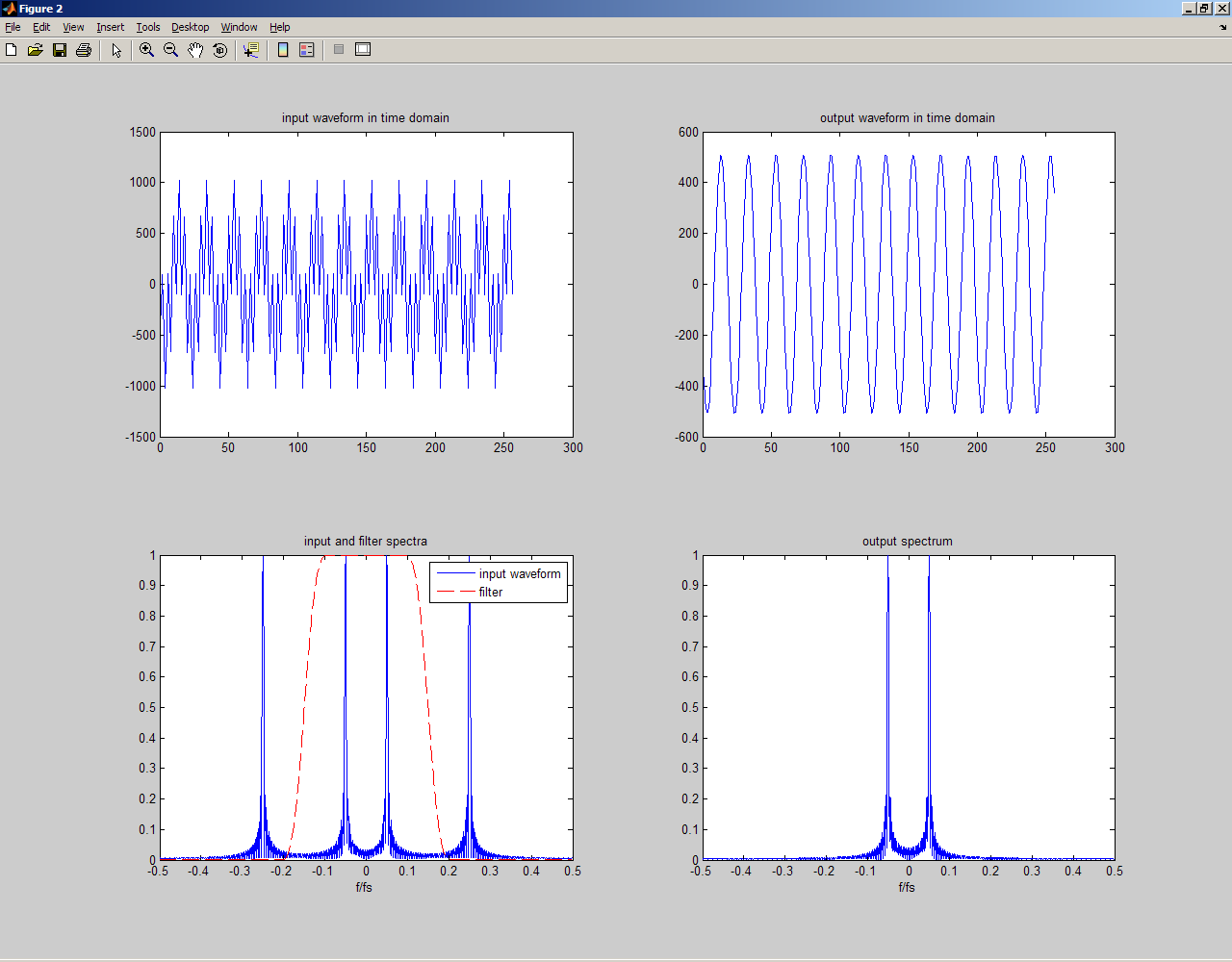
***Шаг 2: Создание файла-сценария M-File***

*В этом разделе будем вызывать Simulink-модель в пакетном режиме при помощи файла сценария MATLAB (M-file). M-file варьирует частоту двух входных синусоид. Результат моделирования сохраняется и выводится на график в конце моделирования.*

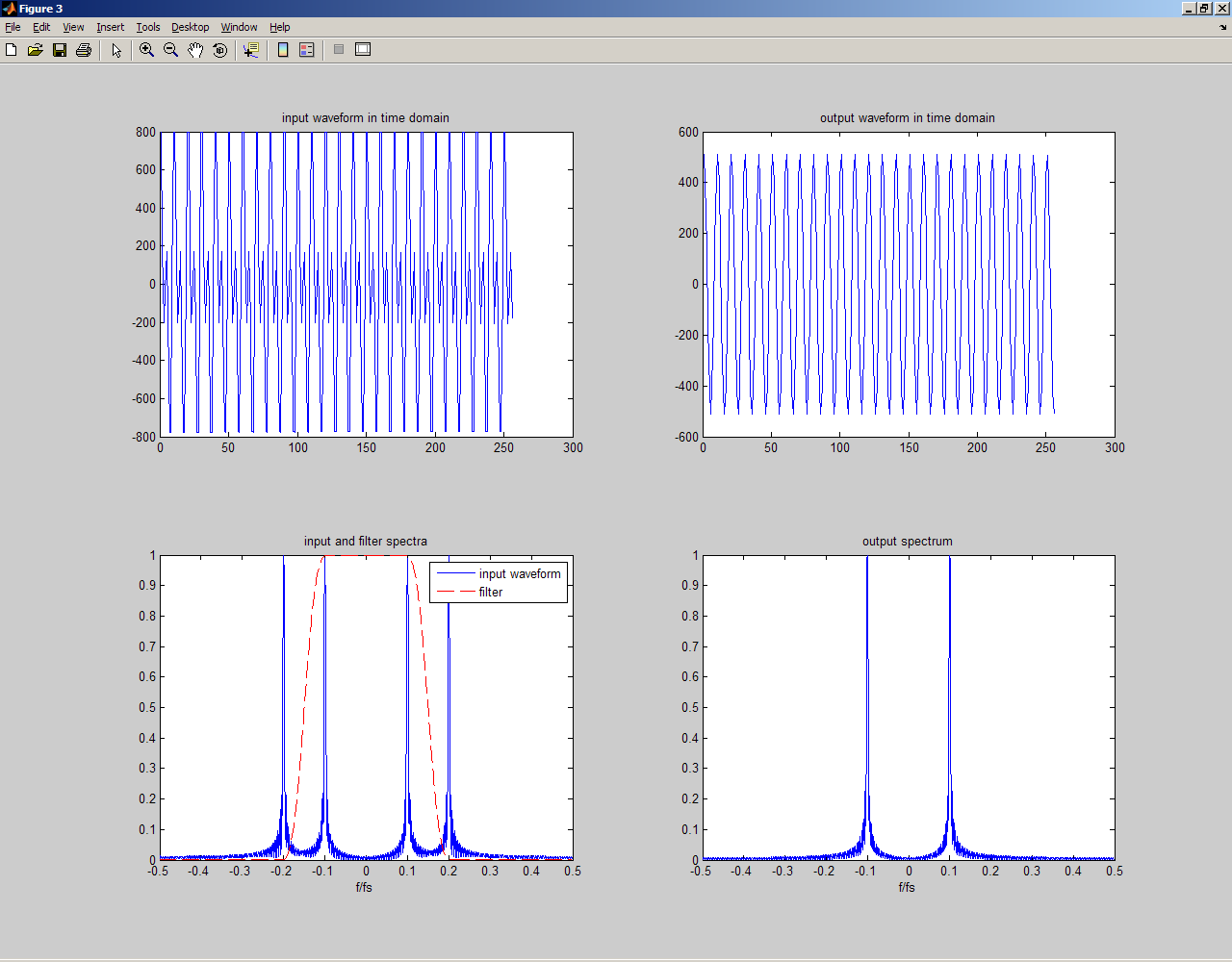
1. Откройте файл **FIR\_32\_testbench.m** и проанализируйте код находящийся в M-file..
2. Запустите FIR\_32\_testbench.m.
3. Пронаблюдайте серии графиков и прокомментируйте результаты работы FIR фильтра. (Вы должны увидеть следующий набор графиков.)



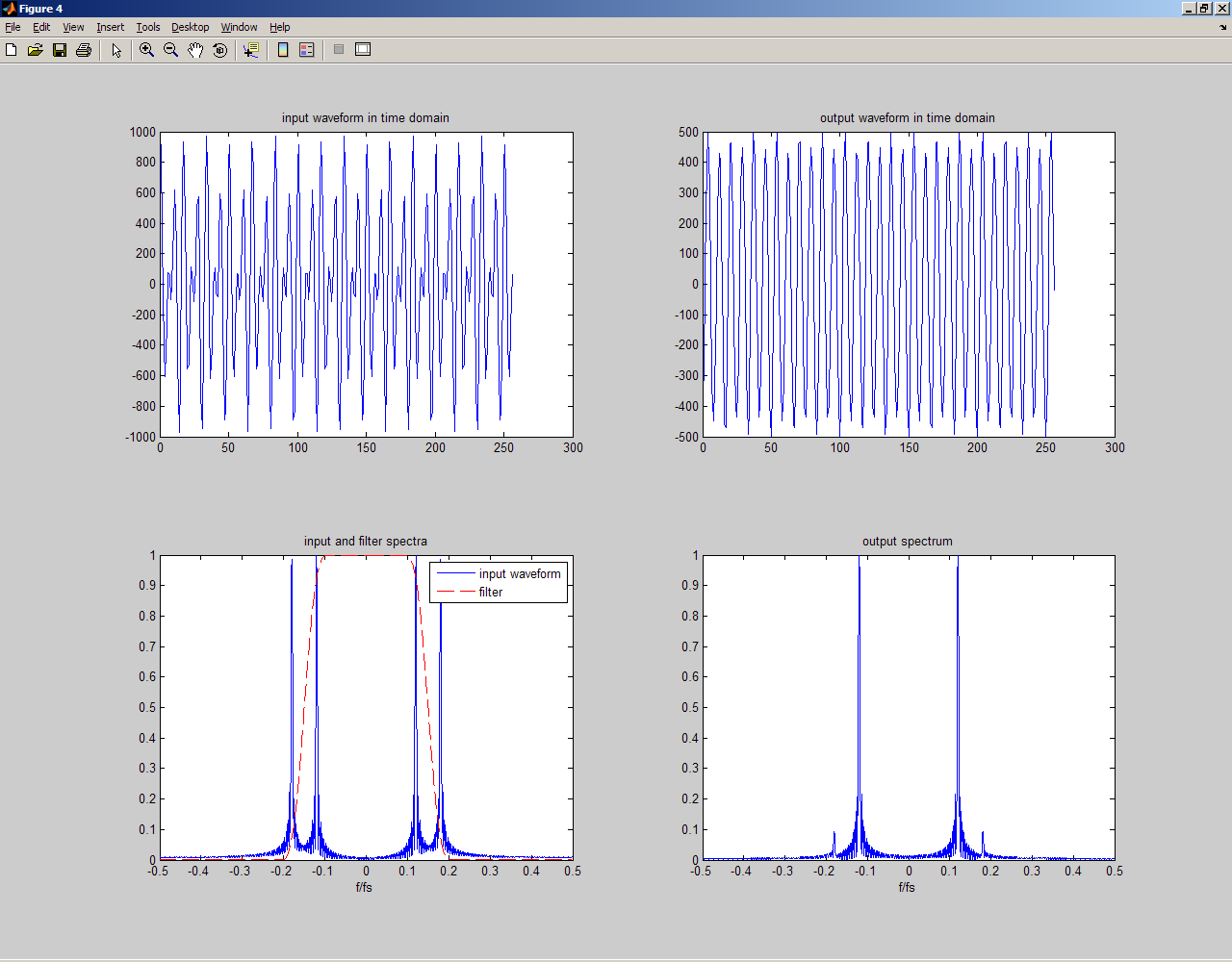
Результат моделирования в условиях и .



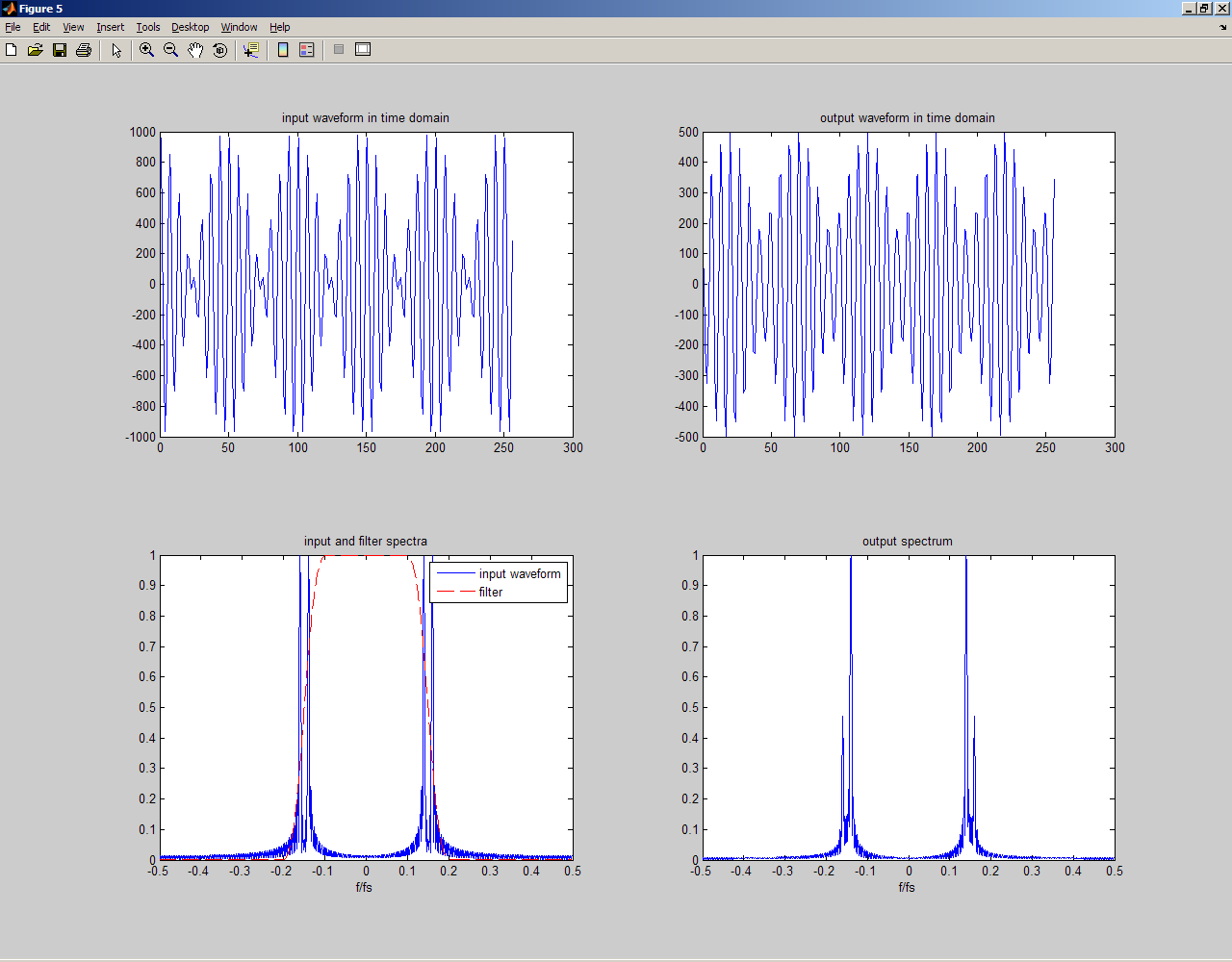
Результат моделирования в условиях и .



Результат моделирования в условиях и .



Результат моделирования в условиях и .



Результат моделирования в условиях и .

Обобщение лабораторной 4

В данной работе получены следующие навыки:

* + Добавление в проект мегафункции Altera’s IP Megacore
  + Интеграция мегафункции Altera’s IP Megacore с проектом
  + Написание файла сценария Matlab M-file для проведения системного тестирования проекта